The Patent Office

INVESTOR IN PEOPLE

09/646796

5

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP9 1RH

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Signed Duncan

Dated 4/5/99

Patents Act 1977
(Rule 16)

The Patent Office

24 MAR 1998

THE PATENT OFFICE
A
24 MAR 1998
RECEIVED BY FAX

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

The Patent Office
Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference

3IMA890 E340756-1 010002
P01/7700 25.00 - 9806843.1

koa.130.uk

2. Patent application number
(The Patent Office will fill in this part)

9806843.0

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

KAL
John Cotton Building
Sunnyside
Edinburgh
EH7 5RA

Patents ADP number *(if you know it)*

If the applicant is a corporate body, give the country/state of its incorporation

7404717001

4. Title of the invention

Software Application Development System

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

Kennedy & Co.
Station House
34 St. Enoch Square
Glasgow
G1 4DF

Patents ADP number *(if you know it)*

7233440001

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

Patents Form 1/77

9. Enter the number of sheets for any of the following items you are filing with this form. Do not count copies of the same document

Continuation sheets of this form	
Description	47
Claim(s)	
Abstract	
Drawing(s)	

10. If you are also filing any of the following, state how many against each item.

Priority documents	
Translations of priority documents	
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	
Request for preliminary examination and search *(Patents Form 9/77)*	
Request for substantive examination *(Patents Form 10/77)*	
Any other documents *(please specify)*	

11. I/We request the grant of a patent on the basis of this application.

Signature Kennedy & Co. Date 24 March 1998

KENNEDY & CO.

12. Name and daytime telephone number of person to contact in the United Kingdom

Neil McKechnie - 0141 226 6826

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 0645 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

SOFTWARE APPLICATION DEVELOPMENT SYSTEM

This invention relates to a software application development system especially but not exclusively for use with computer based cash transaction devices such as automatic teller machines, kiosks, electronic point of sale systems and home PC's.

The use of electronic cash machines for the above and similar purposes is a rapidly growing market. There are many different hardware manufacturers providing this type of equipment and each manufacturer will generally have systems that operate in a different manner. This poses problems for users of the systems such as banks as, if they wish to use equipment from more than one manufacturer then they will require to have the same basic applications specifically adapted to run on each system. Similarly different users using the same manufacturers hardware and having the same basic application requirements will wish these applications to operate differently in accordance with their own preferred customer presentation requirements. Changes in equipment also necessitate a redevelopment of applications. Such a multiplication of systems

inevitably introduces potential areas of breakdown. An increasing reliance on such systems means that system reliability is of the utmost importance.

It is an object of the present invention to provide a system that obviates or mitigates these disadvantages.

According to the present invention there is provided a software application development system especially but not exclusively for use with cash transaction terminals comprising a software platform for development of functional applications for said transaction terminals, said software platform having means for interacting with a plurality of distinctive hardware systems and having customisable means for presentation of said functional applications to cash transaction terminal users.

The cash transaction terminals may, for example, be automatic teller machines, kiosks, electronic point of sale machines, home or office PC's and the like.

Preferably the system makes use of the WOSA XFS standard for interacting with different hardware systems.

Preferably also the system uses a web browser as an application package.

Preferably also the system includes support for a plurality of communication standards.

Preferably also the system includes standard transaction objects or application functions for standard cash transaction terminal functions.

More preferably said application functions can operate independently and communicate with different user interfaces.

Preferably also the use of a web browser provides support for software distribution and network connections.

Specific embodiments of a software application development system in accordance with the present invention are described in the following technical description.

Introduction

Kalypso is a "middleware" software platform from KAL for ATMs, kiosks, EPOS systems and Home PCs. It is an enabling technology that allows high performance mission-critical applications to be developed for these rapidly growing markets. Kalypso allows a single application to be developed to run across all of these delivery systems with built-in support for Internet/Intranet access. Kalypso combines the best of Net technology with multimedia, transaction processing and self service expertise to deliver a state-of-the-art software platform.

Kalypso is an OEM software product and is targeted for use by major vendors such as IBM, NCR, SNI, Diebold, Fujitsu/ICL, Bull, HP/Verifone etc. It is a highly open architecture that allows various levels of customisation and extensions to be carried out as necessary. Kalypso has a carefully thought out architecture that allows various experts to interact with the system in a safe and productive manner. For example, it is possible for the application development to be split-up between application experts, multimedia specialists, systems engineers etc such that large mission-critical applications can be designed quickly, efficiently and above all robustly.

We have also designed a business model for the Kalypso environment with great care. This allows the different organisations participating in the development of a final system to protect their IPR[1] and all reusable software

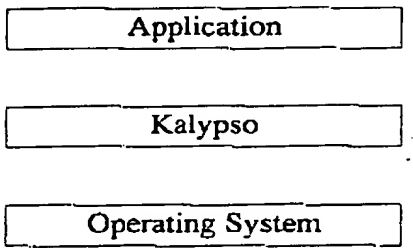
[1] Intellectual Property Rights

components, while continuing to provide the end-user customer with the flexibility that he needs.

Sections 1-4 are intended as an overview of the Kalypso concept.

The Software Architecture from 30000 ft

Kalypso is a middleware layer that essentially sits between the Operating System and the application[2].



This is the view at "runtime" on the kiosk/ATM. At development time, Kalypso provides various tools such as a hardware simulator that allows applications to be developed quickly and easily.

Kalypso currently supports two operating systems - Windows NT and OS/2 Warp. This document will describe the features from the Windows NT perspective.

If you look at Kalypso from the bottom-up, Kalypso is in many ways an Operating System (OS) itself. Having said that, it *isn't* an OS in that it runs on Windows NT and does not duplicate anything that NT itself does. Instead Kalypso extends NT for the kiosk arena, fully utilizing all of NT's features.

[2] Reality is a little bit more complicated than this.

On the other hand if you look at it from the top-down, Kalypso is almost an application - a complete ATM cash application can be created in a few minutes by combining top-level Kalypso components (known as Wizards) together. However, Kalypso is *not* an application - it provides the building blocks required to create a standard application or a fully customised application. Kalypso applications are normally run within a Web Browser (such as Microsoft's Internet Explorer for Windows NT)[3].

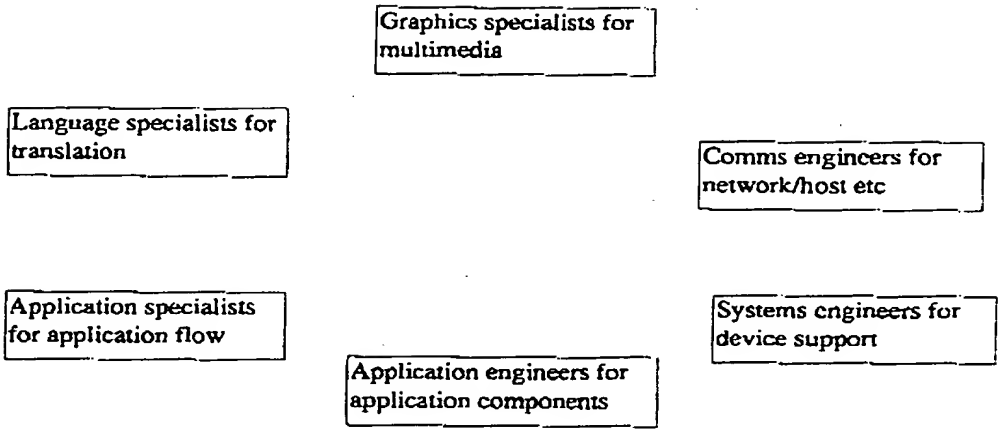
You might wonder whether it is a canned application with some customization - well it isn't that either. Kalypso provides a multi-layered open interface that allows application development to be carried out at the deepest level that you wish to. Systems engineers can extend the software at the lower levels, grappling with issues such as multi-threading, concurrency and re-entrancy, while graphics designers can design the application at the top levels by combining graphics, video and sound.

The openness and extendibility described above does not mean that you lose control of the application when you deliver it to your customer - we have designed a careful business model to allow components to be "owned" by different organizations and or different team members so that issues such as warranty, support, IPR[4] etc can be held in a distributed manner providing maximum flexibility and a "win-win" scenario for all participating organizations. The following is an example of the types of division of responsibility that is supported by Kalypso.

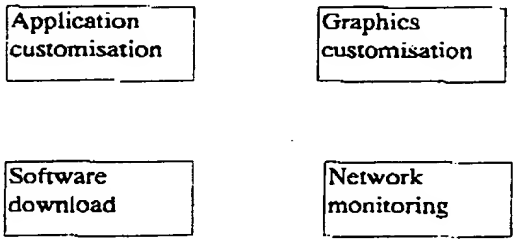
---

[3] Internet Explorer is run in "kiosk mode", which is full screen mode *without* title/menu/status bars etc.

[4] Intellectual Property Rights



Each of the above may be drawn from the same organisation or may be drawn from third party companies as necessary. In addition, the customer's own technical dept may be involved in some of the above, as well as the following:



Kalypso has been designed with the real world in mind. We are aware that expertise is distributed and that a wide range of experts are involved in the creation and maintenance of a mission-critical kiosk application.

Key features

Kalypso provides you with a wide range of features that will enable you to create outstanding applications for the ATM/Kiosk/EPOS/Home PC market.

• WOSA XFS support	WOSA XFS is an open standard from the BSVC (Banking Systems Vendor Council) that provides vendor independent access to hardware on a kiosk or ATM. Full support for WOSA XFS means that a Kalypso application can be ported to various hardware platforms with relative ease.
• Self service support	Kalypso is designed for self service use. KAL has substantial expertise in developing systems software and applications software for ATMs, Kiosks and EFTPOS terminals. Kalypso gives the highest priority to issues such as software robustness, availability, low maintenance, remote monitoring and crash recovery.
• Object-Oriented API	An OO interface provides access for systems programmers to extend the systems software. This makes it easy to add new or non-standard devices with or without help from KAL.
• ActiveX device interface	Under Windows NT, ActiveX components provide the basic building blocks for creating applications within a web browser (MS IE). The Kalypso device interface gives maximum flexibility for expert programmers to fully exploit all hardware features.

• ActiveX Wizards	Wizards implemented as ActiveX controls provide configurable "chunks" of application that can be quickly combined together to create final applications. Wizards can be customized with application-specific graphics and multimedia very easily.
• Concurrency	Kalypso has a fully concurrent architecture. All devices including host access and the screen display can be accessed in a concurrent (ie fully parallel) manner.
• Multi Processing support	Kalypso has been designed with multi-processing and multi-threading applications in mind. Kalypso is thread-safe and supports multiple co-operating applications.
• Multimedia support	Full support for all multimedia features such as video, sound, graphics is provided in a hardware independent manner by exploiting Windows NT support for these features.
• Internet support	Kalypso applications are hosted within a web browser such as Microsoft's Internet Explorer[5]. Full Internet functionality is made available by this choice.
• Intranet support	Use of a browser means that Intranet support is also built-in.

[5] Kalypso applications can also be written independent of a browser if wished. Applications can be written completely in C, C++, Visual Basic etc drawing upon the Kalypso APIs, if preferred. Kalypso also supports Authoring tools such as Icon Author from Aimtech Corp.

• Standalone support	Applications can be created using this technology *even if an Intranet connection is not available*. This allows applications to be Intranet-ready from day one. The complete application would reside on the local hard disk in that case.
• Legacy comms support	Kalypso allows legacy comms to be supported <u>concurrently</u> with Intranet/Internet support. This means that there isn't an either/or choice between Intranet technology and connection to Legacy systems.
• Generic comms support	Kalypso provides support for a legacy comms in a generic way using an Open Standard. This means that applications can be developed independent of the actual legacy protocol(s). This Intranet compatible generic messaging protocol can be customized and extended as necessary. This allows applications to be created that are not just independent of the legacy protocol[6], but can migrate to Intranet/Internet transactions when the Host is ready to do so!
• LAN/WAN support	This allows various connection topologies to be used while still maintaining Intranet-ready status (even in standalone mode).

[6] Protocol independence is at the "Application Layer" level of the OSI model providing **maximum** isolation.

- Robustness

A range of technical features provide robustness for mission-critical applications. Our design philosophy maximises the probability of software errors being detected as *early as possible* at development/test time rather than at runtime in the field. This applies not just for systems software but also for application software.

- Openness

Kalypso has a fully open architecture. All API's will soon be publicly available. Kalypso implements support for a range of open standards including:

- WOSA XFS
- ActiveX
- HTML and various Internet standards
- OFX
- SNMP

Open development tools (Internet tools etc).

- Client-Server support

Kalypso was designed with support for client-server issues between the ATM/kiosk and the backoffice in mind. Kalypso supports transaction technologies such as Microsoft Transaction server and messaging protocols such as OFX.

- Transaction monitoring

Kalypso provides support for Transaction monitors such as NCR's TOPEND. This provides guaranteed transaction commits and load balancing for large systems.

• Use of Internet development tools	Kalypso applications are developed using Internet Authoring Tools. There are a very wide range of tools available and this is an area that is evolving very fast. For the NT/IE environment, our recommended choice is Visual Studio combined with Front Page from Microsoft. However, any Internet tool set can be used due to Kalypso's support for Internet standards. Applications can be written using C, C++, Java, Visual Basic, Javascript, VBScript etc. Java applets may run side by side with ActiveX components.
• Visual development	The development environment is based on Internet tools as mentioned above and is highly visual and interactive.
• Development time environment	At development time, Kalypso provides support via tools such as simulators to allow applications to be developed and tested on development PCs.
• State of Health monitoring	Kalypso provides support for SNMP and remote state of health monitoring by interfacing with standard systems.
• Software Download	Kalypso provides support for Remote Software Download by interfacing with standard systems such as SMS.

As you can see, in many ways, Kalypso is an Operating System. However, it does not duplicate any features of Windows NT. Instead, Kalypso provides the operating

environment that is needed for creating robust kiosk/ATM applications. It does this all the way from the lowest level up to the top level of the application - this means that you may choose to change or customize it as little as you want or as much as you want. We call this strategy an ASOS - Application Specific Operating System.

Kalypso benefits

To OEMs

- Kalypso provides an outstanding environment for creating kiosk/ATM/EPOS/Home PC applications in an Open and Flexible manner.
- Kalypso's technical achievements put it at the forefront of kiosk technology. It's support for WOSA XFS means that Kalypso can provide support for ATM vendor hardware quickly and easily by leveraging off their own WOSA XFS service providers.
- KAL's business model allows the OEM to gain access to this software with no upfront investment. License fees are due only when Kalypso based systems become live at your customer sites. KAL has a simple license fee structure based on a one-off charge per kiosk/ATM.
- Kalypso's ability to provide support across a range of markets, means that you need to train your engineers in only a single set of application tools.
- Kalypso's ability to create Intranet-ready applications *even when there isn't an actual Intranet/Internet connection* means that there is only one application environment that needs to be considered - the Intranet environment.

To end-user customers (eg Banks etc)

- Access to the latest and best technologies combining Intranet/multimedia/PC technologies with Self Service/transaction processing technologies.
- Banks can develop a single Intranet-ready application to run across all their kiosk/ATM/EPOS/Home PC environments independent of hardware differences and independent of the type of comms connection(s) being used.
- Stand-alone kiosks as well as LAN based kiosks can use the same Intranet technology even if only legacy comms is being used. This allows Banks to migrate to new technologies at a comfortable and evolutionary pace.
- Banks can choose to take as much control or as little control of application development as they feel comfortable with, in co-operation with you.
- Banks can setup a dual supplier strategy without being forced to maintain separate applications.

Concepts, Ideas and Know-How

This section details the concepts that we wish to patent and protect.

Top-level Concepts - The Environment

We set out the top-level concepts, including those ideas that are public domain, in order to set the context.

Concept	Description
1 Use of PC Tools	Historically, ATM software applications

	have been created using custom tool sets developed by the hardware vendors specifically for their own hardware. Our idea was to allow generic PC software development tools to be used in the ATM environment. (Most new ATMs are PC-based). As general tools were not designed with the special features of ATMs in mind (eg cash dispensers), Kalypso provides the missing functionality through an object library. (This idea is likely to be public domain).
2. PC Tools - usability	Generic tools from companies like Microsoft have outstanding usability features. Kalypso allows these tools to be used on an ATM bringing the advantages of the sophisticated PC software tools market to ATMs. (Public domain?).
3. PC Tools - features	PC tools have many more functional features than existing ATM-specific software tools. This brings substantial new software features to the ATM market.
4. Use of a Web Browser	Kalypso uses a web browser (such as Microsoft's IE) as the "container" for the ATM application.
5. Web-ATM	Use of a browser allows ATMs to be web-enabled.
6. Web tools for development	Use of a browser means that ATM applications can be created using web tools such as Microsoft FrontPage. Web tools attract massive investment from

	the software industry and acquire new features rapidly. Kalypso allows these technologies to be applied to ATMs.
7. Application has access to various technologies	The top level application can be written using the Kalypso ActiveX controls plus external components from other vendors. These external components can be written using ActiveX Controls, Javabeans, Java applets, VBscript, Javascript or HTML. (Public domain).
8. "COM" based architecture	Kalypso conforms to Microsoft's COM architecture for software components. This makes it easy to add new components from different vendors in creating the final application. (Public domain).
9. Use of WOSA XFS	Kalypso uses the WOSA XFS open standard to support the ATM hardware in a vendor-independent manner. (Public domain).
10. Web-WOSA	Kalypso is the world's first ever implementation of a WOSA XFS compliant web-ATM. (Public domain?). IBM press release 27th August 97.
11. Multi-Platform support	The WOSA XFS compliance allows Kalypso to provide hardware independence. this means that a Kalypso application can be ported to different hardware platforms with relative ease. (Public domain).
12. Support for Legacy host computers	ATMs need to interface with remote host computers of various vintage. Kalypso will provide host-independent support for ATMs using the OFX open standard. (The OFX standard is public domain for PCs, but has never been used for ATMs.

	The standard needs to be adapted for ATM use).
13.Generic comms support	Support for OFX means that Kalypso applications are independent of the peculiarities of the host interface. This allows a Kalypso application to be ported to a new host environment with relative ease. (Public domain).
14.Status monitoring	Kalypso uses the SNMP open standard to monitor the status of the ATM. Alerts and status messages can be sent to a remote monitoring station in this way. (Public domain).
15.Software Distribution	Kalypso's support for web-browser's means that the browser's ability to download content from remote servers can be used for electronic software distribution. Kalypso will also interface with formal software distribution systems such as the web-push technologies and non-web oriented systems such as Netview and SMS.
16.ATM can be connected to the Internet, an Intranet or an Extranet	Kalypso applications can run with an Intranet or Extranet connection or indeed with a connection to the Internet itself. (Public domain).
17.ATM can run "standalone"	The use of browser technology does not mean that a Kalypso applications are *dependent* on a web connection. The application can be fully loaded on to the ATM hard disk allowing the ATM or kiosk to run "standalone". (Public

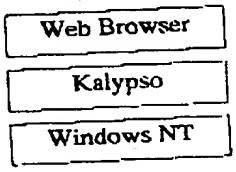
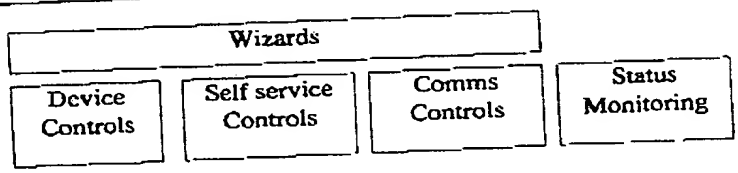
	domain).
18 Robustness	Robustness is of crucial importance for ATMs and kiosks. Kalypso uses a range of public domain ideas for robustness. In addition, the following unique idea enhances robustness: The Kalypso middleware requires applications to interact with it in a well defined way. Even the tiniest transgression from this is detected and flagged by Kalypso as a "fatal error". Fatal errors result in the current environment being abandoned and the application is killed-off by Kalypso. This means that even minor errors in the application can result in a major error condition. Most software components do not do this, but instead choose to ignore what is seen as minor errors. Our view on this is that that allows application errors to survive undetected within applications. The result of our strategy of escalating errors to maximum seriousness, is that errors are found early at development time or test time and are never allowed to reach a live environment. On the otherhand, this technique potentially runs the risk of a fatal error being reported in the field for what might be

	considered a non-serious situation. However, in return for taking this risk, the application achieves outstanding robustness as all errors tend to be detected before the software goes to the field. The technique of detecting errors in this way is called "assertion" in software engineering. This is a public domain idea. However, our contribution is to assert absolutely *all* disallowed cases whether serious or not. This results in a highly robust final application. (At KAL we regularly achieve zero defect software releases even for large systems - a rare occurrence in the software industry).
19.Security	The first level of security is via digital certificates that are carried by all our ActiveX controls. (Public Domain). In addition each Kalypso control can be turned on or off using the security control. Please see the next section for details.
20.Openness	Kalypso uses various open standards to provide an open environment for developers. In addition, Kalypso can be accessed at several levels depending on the level of control required by the application developer. At the top-level a developer can use the Kalypso wizards. Alternatively, if the wizards do not

	provide the exact features required, the application can directly access the device controls etc.
21.Client-server support	Kalypso allows robust client-server applications to be developed, with Kalypso running on the client. (Public domain). See next point.
22.Scalability	Kalypso interfaces with Transaction Processing (TP) Monitors such as Topend from NCR and three-tier TP servers such as MTS from Microsoft to allow scaleable[7] client-server systems to be developed. (Public domain).
23.ATM simulator	Kalypso incorporates a simulator for ATM hardware that allows Kalypso applications to be developed and tested on a PC. (Public domain).
24.Development environment on an ATM	The Kalypso development environment can be run on an ATM, making it easier for applications to be developed and tested. (Public domain).
25.Unique combination of concepts	Of the top-level concepts listed above some are in the public domain, and others not. The combination of these concepts to form the final Kalypso concept is believed to be unique, and not in the public domain.

[7] Scaleable means that very large networks of client ATMs running Kalypso can be created. These large networks would have multiple servers. The TP monitors allow clients to be matched up with available servers dynamically and in a fault tolerant manner.

Top-level Concepts - The Software Architecture

Concept	Description
1. Kalypso is a middleware component	 Kalypso sits between the web browser and the operating system. It provides the ATM-specific services needed by the browser to run the end-user application.
2. Kalypso consists of Five sub systems	 **The bottom layer** • The device controls provide hardware independent access to the special devices on the ATM or kiosk. • The self service controls provide functionality that is necessary for creating self-services applications. • The comms controls provide access to the remote host computers using the OFX standard. • The status monitoring system monitors

	the health of the ATM or Kiosk and sends status and alerts to an external monitoring station using SNMP alerts. **The top layer** • The wizards are "transaction objects" that implement commonly used transactions on ATMs. (eg dispense cash - known as "fastcash", print a statement etc).
3.Use of "COM"	All sub systems are implemented as a set of "COM" components with an ActiveX interface. (Com and ActiveX are Microsoft technologies. ActiveX components are web-enabled, are downloadable by a browser, publish their interfaces, and can manage software versioning). These are published Microsoft technologies.
4.Wizards	Kalypso wizards are a fundamentally important concept within Kalypso. A wizard is a "chunk" of re-usable functionality. It is part of an application. For example there will be a Kalypso wizard that creates a statement print and another one for fastcash[8]. The wizards have a unique and powerful architecture: A wizard is implemented as an ActiveX object that is responsible for a transaction. When the application enters

[8] The wizards are currently under development

a wizard, the wizard takes full control - ie the wizard *is* the application while it is running. An application would consist of several wizards.

Wizards encapsulate all of the features and functionality required by that particular transaction (or chunk of application). However, the wizard is completely independent of the user interface. That means that the visual appearance and the way of interacting with the wizard can be changed from application to application without affecting the self service knowledge within the wizard. The encapsulated features within the wizard can be reused between different applications even though the look and feel could be completely different.

ActiveX technology was found to be ideal for implementing wizards. Kalypso wizards interface use the Kalypso device controls and the other sub systems (comms, self service etc) and encode all of the top-level control logic within the wizard. In order to separate out the user interface, the wizard outputs it state as a set of ActiveX events, and receives input via ActiveX properties and methods.

What this means is that a Kalypso wizard can be placed on a browser page. It is invisible to the user at this stage, but is able to perform the ATM transaction. Using web tools, a web designer can now build the user interface around the wizard, without any risk of compromising the self service knowledge within the wizard. The designer can layout buttons, graphics, hotspots, videos as he wishes. he can make these buttons send messages to the wizard, and can make the visual content react to the events emanating form the wizard. The technology needed to connect up the wizard to these user interface elements is standard to web browsers.

Now this means that the ATM transaction can look completely different between two banks, for instance, even though the critical ATM code is exactly the same. For instance, one bank could have a single button that says dispense $10 and have no other user interface, and allow the ATM to dispense the cash, while the other bank could have multiple graphics and animations and videos which run concurrently and react to the different stages within the wizard. (eg when the wizard is counting the cash, it could run an animation showing the cash being counted, and when the cash is presented,

it can stop the animation and run a video of someone saying "please take your cash"). Even though the look and feel of the application is massively different, the underlying logic would be exactly the same, would be reusable between applications, and would be very robust - as it needs to be written only once and is used again and again.

A huge benefit to banks is that the look and feel of the application can be changed regularly by a graphics person without risking the "correctness" of the application.

The benefit to developers is that new applications can be created very quickly and easily and can have a completely different look and feel.

As the wizards are built on the device controls and comms controls, it means that these wizards can be reused on different hardware platforms and with different host connections without affecting the implementation of the wizard.

Wizards have one more essential feature - they are able to interpret the capabilities of the hardware. This means that a Wizard is able to perform a

	transaction using devices with varying capability. For example consider the differences between a motorized card reader and a swipe card reader. In the first case, the wizard must eject the card and tell the user to take the card. In the latter case, the wizard does not need to do that. Kalypso wizards allow applications to run across differing hardware capabilities. This is an essential part of Kalypso's support for Extranets. It allows an application to be run on non-homogeneous networks of kiosks and ATMs.
5. The Device Controls	The device controls provide hardware independent access to the special devices on the ATM or kiosk. Each device control acts as a persistent server that can be controlled and interrogated by one or more applications. The device control abstracts the details of the hardware underneath it and acts as a complete server for that device. The application(s) interact with the device control via an ActiveX interface which is scriptable. (ie a few lines of VBscript within an HTML page can control a Kalypso control). The device controls are fully concurrent. That means that they run independent of each other in "parallel".

	This is vitally important in an ATM or kiosk as the cycle time of a transaction is critical. Kalypso allows all devices to be controlled concurrently. This also means that these controls can execute commands asynchronously. ie, the application can ask a control to do something while the application does something else. Once the control has completed it's task, it reports back with the result (success or failure) via an ActiveX events. This event-driven design makes it easy to create applications using browser technology. There are various web tools that provide easy-to-use graphical interfaces for creating event-driven applications.
6.Self Service Controls	The self service controls provide functionality that is necessary for creating self-services applications. These controls have the same "server" architecture as the device controls and are able to execute commands asynchronously. Each of the controls is described in more detail in section 0.
7.Comms Control	The comms controls provide access to the remote host computers using the OFX standard. These controls have the same "server" architecture as the device controls and are able to execute commands asynchronously.
8.Status Monitoring	The status monitoring system monitors the health of the ATM or Kiosk and sends

	status and alerts to an external monitoring station using SNMP alerts.
9.Extranet	A very important feature of Kalypso is it's support for Extranet based applications. An Extranet is a network connecting the Intranets of two or more companies. Consider a scenario where a Bank and an Airline choose to co-operate. The Bank has a network of ATMs (say) and the Airline has a network of ticketing kiosks. Bank Airline ATMs Kiosks Now consider that Kalypso is running on the kiosks and the ATMs. The Bank can run the Airline ticketing application on the Bank's ATMs by simply setting up a web link from the Banking application to the Airline application. Similarly the Airline can run the banking application on it's kiosks. Now consider the fact that the kiosk and the ATM have different hardware types and capabilities. Kalypso's support for WOSA XFS takes care of the vendor dependent hardware differences. Kalypso's support for "capabilities" (see below) takes care of the capability

differences.

The support for capabilities means that the application can interrogate the device controls as to their capabilities. That means that banking application (for instance), when running on the airline kiosk knows it cannot dispense cash, but can offer most other banking services (eg statements, balance enquiry etc).

Similarly, the Airline application can allow the user to buy an E-ticket (say) and pay for it, but the application knows that it cannot dispense a boarding card (say).

Now consider what happens when other partner companies join the extranet. For instance a supermarket and an insurance company might join. This would allow the combined set of e-services of the partners to be made available through the combined set of kiosks and ATMs. In order to make this possible there are three important requirements, all of which are satisfied by Kalypso:

- The middleware (ie Kalypso) must be web-enabled
- The middleware must be hardware independent

	• Capabilities must be implemented so that differing hardware capabilities can be managed
10. Capabilities	All Kalypso controls implement a "capabilities" interface. This interface allows the application to interrogate the capabilities of the control as well as the device that it represents. This is a fundamental part of Kalypso's Extranet support. It allows an application to dynamically configure the services that it can provide based on the capabilities of the hardware available on the kiosk.
11. Differentiation and standardisation	One of the business requirements of the OEM's and the end-user institutions that would use Kalypso is the ability to differentiate themselves from the competition. Kalypso on the otherhand provides a standard interface that may at first appear to disallow differentiation. This is not the case. Kalypso allows differentiation with standardisation. This is possible through capabilities. Kalypso is able to provide a superset of features from each hardware vendor using the capabilities interface to let the application decide whether it wishes to use the enhanced features. This means that new features can be added to Kalypso without requiring a "least-common denominator"

	approach. Standardisation is provided by ensuring that there is only one way to access any particular feature. That means that all common features have a single API (therefore standardisation) while new features can be added without making the application dependent on it (due to the capabilities interface) - hence differentiation.
12.Controls run without the device	An important feature of Kalypso controls is that they run on an ATM even if the actual hardware device is not present. ie, the cash dispenser control will run on any kiosk. This allows an application that uses the cash control to startup and run. When it requests the capabilities of the control it will reply that the device is not present so the capabilities are Null.
13.Ignore mode	In the case where the device is not present, the application can request the control to run in the "ignore" mode. In this mode, the control will return "success" for every command. This allows the application to use generic code that does not need to test whether the device is present at each step. This simplifies the code that needs to be written when writing an application to cope with various hardware capabilities.
14.Security	All Kalypso controls include a security mechanism. The purpose of this mechanism is to allow the "Methods" of the

	controls to be enabled and disabled. This is particularly important in an Extranet environment where applications of differing abilities run on the kiosk or ATM. For instance a bank would wish to disallow the airline application (from our example) from dispensing cash. This is done through a key passing mechanism as follows: • The Kalypso security control allows the current security configuration of the ATM/Kiosk to be set. • Using the control, the owner of the kiosk can specify details of the security configuration (ie which "methods" are disallowed and which are allowed). • Applications identify themselves to the security control via a digital certificate. This sets the security configuration as specified by the ATM owner. • If the application attempts to call a method of a control that is not allowed, a trap is generated. This allows control to pass to the ATM owner's application for cleanup.
15. Concurrency	We mentioned that the Kalypso controls support concurrency. This means that all the controls can run independent of each other in parallel, and are able to execute commands asynchronously.

16.Multi-processing support	The Kalypso architecture allows multiple applications to run simultaneously. This means that two or more applications can use the Kalypso interface without causing problems.
17.Windows CE	The Kalypso controls have been designed to allow them to be migrated to Windows CE. This would allow a single application to run on CE,95 and NT based kiosks and ATMs.
18.Multi-platform support	The ability to run on various operating systems and hardware means that a very wide range of kiosks and ATMs can share the same application. From a high-end ATM to a low-end portable kiosk, Kalypso provides a standard interface to web-based delivery of e-services.

Software Design Concepts

Concept	Description
1.Activex can be replaced with Java beans.	Although the controls have been designed using Microsoft's ActiveX model, the underlying architecture is not dependent on ActiveX. Each Kalypso component is created as a C++ object which can be accessed by a light-weight ActiveX "hook". This means that the ActiveX hook can be replaced by a Javabeans hook thereby providing the same features using the Javabeans component model.
2.Persistence	One of the problems faced by a web

	designer is that objects placed on a web page have a very short life span - they survive to the end of the page and are destroyed when the page is changed. The ActiveX "hook" idea solves this problem for us. The underlying Kalypso object remains persistent while hooks on each page access the object. This means that the ActiveX control appears to be persistent to the application from page to page. Lack of persistence provides an additional problem for the application developer: the storage of application-wide data. The Kalypso scratchpad control solves this problem. (See later).
3. Error handling	One of the complications with error handling in ATM applications is that components can return a large number of error cases. Handling all these error cases immediately can lead to very complex code. Kalypso helps with this process by separating the responses it send to the application into two categories. "Good responses" and one error response. Most commands normally have a single good response and a single error response. Some commands can have more than one good response. The definition

	of a good response is a response that allows the transaction to continue forward. In a bad response case, the current transaction flow would normally have to be aborted. In this case the control flow would jump out of the normal flow process to handle the error situation. This means that the normal flow is not cluttered by handlers for each of the error cases that can occur at each stage. Control can be transferred to generic error handlers which can either attempt to recover from the error or abort the transaction completely. This design allows the application code to remain as clear and concise as possible while also encouraging the developer to handle all error cases by calling an error handler.
4, Fatal error handling	As we mentioned earlier all illegal and disallowed cases result in a fatal error exception being generated. This results in robust applications. The actual way in which the error is handles can be set by the application developer. The fatal error can be returned to the application for handling or it can be used to reboot the ATM. In the development environment fatal errors result in a message box being displayed.

5.Linear code	In addition to the traditional way that ActiveX fire events to the container, Kalypso device and self-service controls are able to queue up events and return them one by one when requested. This allows C++ applications to be written in a linear fashion rather than an event driven fashion. The ability to write linear code makes it much easier to develop and maintain the complex logic required in self-service applications.
6.Self Service Controls	The self service subsystem consists of the following controls:
• Watchdog Control	The watchdog runs in a separate NT process and reboots the ATM if the application crashes. It does this by requiring the application to constantly poll it saying that it is ok. The watchdog control can also be used to daily reboot the ATM.
• System Escape Control	The system escape control is used to reboot the ATM. It has a vendor specific "exit" which allows the hardware vendor to modify it's behaviour. The system escape control ensure that cached data (eg in the DataCollect control and the Trace control) are flushed to disk before rebooting.
• DataCollect Control	This allows the application to collect raw data for statistics. The application is able to log various events into this control. Each event is timestamped and stored on the hard disk for future

	reference. As disk writes can take time, this control logs to memory but store on hard disk only when the ATM is idle. The storage for the DataCollect control is held in a fixed size disk file. This file is allocated at bootup to ensure that disk usage remains constant throughout. The file is used as a ring buffer. So, if the file size is set to 10Mb in size, the last 10Mb of raw data will always be present on the ATM/Kiosk. The collected data would normally be exported to a remote location for analysis.
• Trace Control	The trace control is similar to the DataCollect control except that it is used to collect trace events of the application and the underlying middleware. This is targeted mainly at the application developer to allow the software to be debugged. Normally tracing is turned on only when a problem is suspected. However this is a problem as tracing can affect the dynamics of a problem and make it harder to find bugs. Our approach is two fold: one to have tracing always enabled and two to address the reasons why tracing is not normally always enabled: performance and disk usage. Performance problems are eliminated by tracing to memory. The trace is written

to disk only when the ATM is idle. (All ATMs go through an idle cycle between two users. This is a relatively long time even when people are queuing at the ATM).

The disk usage problem is eliminated by using a ring buffer and guaranteeing the file size as for the DataCollect control.

The fact that tracing is always enabled also means that it makes it much easier to catch one-off problems. Normally tracing is enabled after a problem is reported - a bit too late to catch the problem especially if it is a rare event.

In addition the trace control has a special, ATM vendor dependent feature. Some ATMs have a limited amount of non-volatile RAM. The trace control writes the most recent trace information to this RAM in a ring buffer fashion. As the RAM is quick to access this does not cause a performance problem. However, if the ATM freezes up or crashes for an unknown reason the RAM contains the trace of what happened just before. (In the case of a freeze of this sort[9], any unflushed trace data would have been

[9] which is usually very rare!

	lost, if not for the non-volatile RAM).
• Scratchpad Control	We mentioned earlier that browsers do not allow objects or data to persist across pages, by default. The scratchpad control on the otherhand has a persistent object at it's core and allows the application to store and retrieve data at any time. The control supports the Vbscript "variant" type which allows all types of data to be stored and retrieved. This control also allows data to be shared between multiple applications. This is done by marking the data as being shared.
• Supervisor Application	Kalypso allows the supervisor application to be run simultaneously as a separate application. This means that on an ATM with a rear screen the operator can interact with the ATM without taking the machine offline. It allows the operator to access statistics etc while there is a user at the front of the machine. If the operator wishes to carry out intrusive maintenance functions, he would then have to take the machine out of service. Thus the supervisor application can run in on-line mode or off-line mode, providing only a subset of features in the on-line case.
• Security Control	This was detailed earlier.
• Registry Control	This control allows the Windows NT registry to be manipulated by the

	application.
• DirectroyTree control	
• AppLauncher control	
• INI file control	This control allows the a Windows INI file to be read from the browser.
• FTP EXE	
• Signature capture control and SP	This control allows a signature capture device to be supported.
• Key capture Control	This control allows special windows keys such as ctrl-alt-del and alt-tab to be captured. This is essential in kiosk applications that have a full PC keyboard.
• Popup suppression control	This control allows popup windows to be monitored and captured. This is useful to allow software components from other vendors to be used in a self service application. Most software is not written to be used in a self service environment. This means that these software components expect to be able to talk directly to the user via popup windows. This is unacceptable in a self service environment. (The main application must have a complete monopoly over the user dialog). This means that a large amount of PC software components cannot normally be used in an ATM/Kiosk environment.

	The Kalypso popup control goes someway in helping with this process. This control monitors popups, and when a popup occurs recognise it, and executes a pre-determined sequence of tasks. (eg hide the popup and press the OK button). The popup control can do this quickly enough so that the user should not notice the popup occurred at all. This allows a large range of software components to be used in creating self-service applications. For example, we recently used a third party fax utility to send faxes from a kiosk. This software would popup to inform the user of the progress of the fax. Our control intercepts the popup and presses the OK button, and notifies the main application of the progress of the popup, allowing the application to deal with the status change.
• Global config file control	
• Telephony control	
• SSMS control	
• Timer control	
• Screensaver control	
• Multiple language control	

• Clock synch control	
7. Windows NT standard printer support	
8. Printer SP - HTML field	
9. Printer SP - Flush	
10. The Kalypso API	
11. Smartcard support	PC/SC support
12.	

Software Implementation Concepts

Concept	Description
1. Asynchronous methods	All device and self-service control methods which are expected to have appreciable duration are implemented asynchronously. This allows the application to carry on other tasks in parallel with the method execution.
2. Commands by name not number	Though based on WOSA, which assigns a different number to each command, Kalypso controls have differently named methods and events associated with each operation.
3. Mapping of WOSA events to set of	A typical WOSA command may, on completion, generate one of 30 to 50

completion events	different events in principle. Dealing with so many events is time-consuming and error prone for the application writer. Kalypso reduces the set of possible outcomes to a small number of completion events which are clearly named: outcomes which can only happen if there is a bug in the application cause a fatal error to be triggered.
4. Small number of completion events - necessary results only	The number of possible completion events for each method is kept to a minimum: this makes it easy for the application writer to develop reliable code rapidly.
5. DeviceError event for all device errors	All methods that involve an action on a device can generate the DeviceError event if there is some kind of hardware failure. Having one event to cover all different types of device error allows the application writer to write one handler which examines the status properties of the controls to determine what action to take: the error handling code can be encapsulated rather than scattered over many event handlers.
6. Status as properties	
7. Capabilities as properties	
8. Event thread for handling async commands	All Kalypso controls create their own thread, called the event thread, when first constructed. This enables asynchronous operations to be carried out: when an asynchronous method is

	called, a command message is sent to the event thread. The event thread carries out the command and sends a message back to the main thread on completion: the completion message causes the appropriate event to be fired. Doing commands using the event thread means the main thread (the application) is free to get on with other tasks in parallel. The event thread also ensures that the device state persists from one application page to another: though the controls on the HTML pages are being continually created and destroyed, the event thread remains running and ensures that the connection to the device is never lost.
9.WOSA session automatically opened	The WOSA session is automatically opened when a Kalypso device control is first used: there is no need to manually call an Open method.
10.WOSA session maintained between pages	See event thread discussion above.
11.Timeouts for interactive methods only	Though all WOSA methods require a timeout to be provided, the timeout is not appropriate or meaningful for the majority of commands. Kalypso requires a timeout to be supplied only where it is meaningful to do so.
12.Cancel for appropriate methods only	WOSA allows a cancel command to be sent after any other commands. In reality it is only possible to cancel certain

	operations: Kalypso provides cancel methods only for those operations which can really be cancelled.
13. Request Ids abstracted out	WOSA returns request ids associated for each asynchronous operation. Request Ids are abstracted out in Kalypso.
14. Reentrant object layer for multithread/ multiprocess	
15. No direct WOSA access	
16. Invisible frame for unsolicited events	Some applications need to be able to react on an event no matter at what stage the event occurs - for instance if the safe door is opened, the application may need to shut down immediately. This is not easy in web base applications because it can mean that every single page needs to contain some code to handle the event. This problem can be solved using frames: the idea is that all visual aspects of the application and the main control flow will be carried out in one frame that fills the whole screen. However a second frame will be constantly running along side the main frame which will take up no screen space. The invisible frame can include all the device controls needed to detect the events that must be reacted on - and this frame can take control and close down the main frame

	(or whatever) when it detects one of the relevant events.
17.	

Software Testing Concepts

Concept	Description
1. Automatic testing	
2. HTML based testing in stress testing mode or user mode.	A test HTML based application has been developed for testing Kalypso device controls. The application allows the operator to select a subset of the devices for testing. For each device, two test sequences are defined: there is a test sequence requiring operator interaction (for entering/removing card etc) and one requiring no operator action. If the latter sequence is chosen, the interaction-free test sequences will be repeatedly run for the selected devices. This allows Kalypso to be easily stress tested. If a more complete test is required, the sequence with operator interaction is chosen and the application leads the operator through a detailed test sequence for the chosen devices. In both cases, testing is as automated and therefore repeatable as possible.
3.	

Further modifications and improvements may be incorporated without departing from the scope of the invention herein intended

PCT/GB99/00927

23/4/99

Kennedy & CO